IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	John K. Vincent et al.
Serial No.:	09/226,939
Filing Date:	January 8, 1999
Group Art Unit:	2162
Examiner:	Anh Ly
Confirmation No.:	8916
Title:	SYSTEM AND METHOD FOR RECURSIVE PATH ANALYSIS OF DBMS PROCEDURES

Mail Stop Notice of Appeal
Commissioner of Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

PRE-APPEAL BRIEF REQUEST FOR REVIEW

The following Pre-Appeal Brief Request for Review is being filed in accordance with the provisions set forth in the Official Gazette Notice of July 12, 2005 ("OG Notice"). Pursuant to the OG Notice, this Request is being filed concurrently with a Notice of Appeal.

In the prosecution of the present Application, the PTO's rejections and assertions contain clear errors of law. Most notable of the legal errors present in the examination of the Application is a failure of the Final Office Action dated June 27, 2008 (the "*Office Action*") to establish a prima facie rejection of at least independent Claims 40, 48, 65, and 72 and dependent Claims 50, 71, 73, and 74 under 35 U.S.C. § 103(a). In this Pre-Appeal Brief Request for Review, Applicants request panel review of the identified claims.

I. Independent Claims 40, 65, and 72

The rejection of independent Claim 40 is improper at least because the proposed *Caron-Colby-Shan* combination fails to disclose, teach, or suggest "stopping the recursive query of the database upon identifying a dependency that is already included in the dependency information tracking array." Similarly, the rejection of independent Claim 72 is improper at least because the proposed *Caron-Colby-Shan-Laursen* combination fails to disclose, teach, or suggest "for each of the one or more dependencies identified, determining whether the dependency already occurs in the graph" and "terminating the recursive query of the database upon determining that one of the one or more dependencies already occurs in the graph." The *Office Action* cites *Caron* for disclosure of

Applicants' recited dependency information tracking array and *Shan* for disclosure of stopping the recursive query. However, the combination of references does not disclose Applicants' recited operation.

Shan merely discloses "a novel fixpoint operator and new transformation procedures . . . for translating a recursive query into a relational algebra expression and then simplifying the expression." (*Shan*, page 12). Page 16, lines 121 and Figure 3 of *Shan*, as identified by the Examiner, merely indicates that recursive relations an expression for a fixed point operator that includes recursive relations. (*Shan*, page 16, paragraph 1). The inputs of the expression, R_1-R_N, are recursive inputs. (*Shan*, page 16, paragraph 1). The outputs are also recursive relations are fed back into the expression as recursive inputs R_{C1} through R_{CK}. (*Shan*, page 16, paragraph 1). The ultimate output is a single relation. (*Shan*, page 16, paragraph 2). Thus, *Shan* discloses that the translated recursive query is "an expression that includes a fixpoint operator." (*Shan*, page 12, paragraph 6; page 16, paragraphs 1-3). However, there is no disclosure in *Shan* of stopping the recursion by freeing with the operator to pose the queries, as suggested by the Examiner. More importantly, there is no disclosure in *Shan* of "stopping the recursive query of the database upon identifying a dependency that is already included in the dependency information tracking array," as required by Applicants' Claim 40. *Shan* merely discloses that the query includes the query and that the query is then "used to express various recursive queries" of a database. Furthermore, even to the extent that *Shan* discloses "stopping the recursion by freeing with the operator to pose the queries," as suggested by the Examiner (a fact that Applicants expressly dispute above), stopping the recursion to pose queries is not analogous to Applicants' recited steps of "stopping the recursive query of the database **upon identifying a dependency that is already included in the dependency information tracking array**," as recited in Claim 40, and "terminating the recursive query of the database upon determining that one of the one or more dependencies already occurs in the graph," as recited in Claim 72. Neither *Caron* nor *Shan* nor their proposed combination discloses this combination of claim elements.

The additional disclosure of *Laursen*, as applied to Claim 72, does not cure the deficiencies identified above. *Laursen* does not at all relate to recursive queries of a database. There is no discussion in *Laursen* of querying a database, no mention of the word "recursive" as used in the context of a query, and certainly no discussion of terminating or stopping a recursive query. The cited portions of *Laursen* only disclose that "the topology of the network [for application servers and clients] is described as a directed graph whose vertices are nodes (either intermediate nodes or endpoints) and whose edges are data links between nodes." (*Laursen*, column 12, lines 35-36 and 40-45). Thus, while *Laursen* discusses generally a "graph," such disclosure is limited to a depiction of a topology of a network of servers and clients. *Laursen* does not at all relate to "for each of the one or more dependencies identified, determining whether the dependency already occurs in the graph" and

"terminating the recursive query of the database upon determining that one of the one or more dependencies already occurs in the graph," as recited in Claim 72.

For at least these reasons, the rejections of Claims 40, 65, and 72 should be withdrawn.

II. Claim 48

The rejection of independent Claim 48 is improper at least because the proposed *Caron-Colby-Shan* combination does not disclose, teach, or suggest "identifying one or more cyclic dependencies among code objects stored in the database." The *Office Action* acknowledges that *Caron* does not disclose the recited claim elements but relies upon *Colby* only for disclosure of querying a relational database and identifying database objects. (*Office Action*, page 11). The Examiner identifies no reference as disclosing "one or more cyclic dependencies among code objects stored in the database," as recited in Applicants' Claim 48.

Even to the extent that *Colby* discloses querying a relational database and identifying database objects, Applicants respectfully contend that *Colby* does not disclose "one or more cyclic dependencies among code objects stored in the database." Rather, *Colby* merely discloses a system for answering a database query where the tables "include foreign key-primary key relationships." (Column 7, lines 4-8). According to *Colby*, the "primary key uniquely identifies each row in a database table" and "can be one value from a single column or a combination of values from multiple columns." (Column 7, lines 8-11). By contrast, "[a] foreign key column contains only the values of a primary key column of another table and establishes a many-to-one relationship between the two tables." (Column 7, lines 11-13). "Unlike a primary key column a foreign key column can contain duplicate values." Thus, the cited portion of *Colby* merely discloses that a primary key uniquely identifies each row in a table but that a foreign key establishes a many-to-one relationship between two tables. Another portion of *Colby* cited by the Examiner discloses a derived table that is created in an example rewritten query. However, that portion of *Colby* merely discloses that the primary key results in an accurate summation. It does not at all relate to "cyclic dependencies," as recited in Claim 48. Accordingly, *Colby* does not disclose, teach, or suggest "identifying one or more cyclic dependencies among code objects stored in the database," as recited in Claim 48.

For at least these reasons, the rejection of Claim 48 should be withdrawn.

III. Claim 50

The rejection of dependent Claim 50 is improper at least because the proposed *Caron-Colby-Shan* combination does not disclose, teach, or suggest "generating a dependency graph for code objects stored in the database based at least in part on the dependency information tracking array." The *Office Action* cites *Colby*; however, the cited portion of *Colby* merely discloses tables that

"include foreign key-primary key relationships." (Column 7, lines 4-8). According to *Colby*, the "primary key uniquely identifies each row in a database table" and "can be one value from a single column or a combination of values from multiple columns." (Column 7, lines 8-11). By contrast, "[a] foreign key column contains only the values of a primary key column of another table and establishes a many-to-one relationship between the two tables." (Column 7, lines 11-13). "Unlike a primary key column a foreign key column can contain duplicate values." (Column 7, lines 13-14). According to *Colby*, the primary key "results in an accurate summation." (Column 10, lines 10-26). However, the use of primary key and foreign key relationships is not analogous to "generating a dependency graph for code objects stored in the database based at least in part on the dependency information tracking array," as recited in Applicants' Claim 50.

For at least these reasons, the rejections of Claim 50 should be withdrawn.

IV. Claims 70 and 71

The rejections of dependent Claims 70 and 71 are improper at least because the proposed *Caron-Colby-Shan-Laursen* combination does not disclose, teach, or suggest that "the specifications of object-oriented code objects comprise PL/SQL specifications for a collection of stored functions and procedures identified as a single entity." The *Office Action* acknowledges that *Caron*, *Colby*, and *Shan* do not disclose the recited claim elements and instead relies on *Laursen* for disclosure of the recited claim elements. Specifically, the *Office Action* states that "*Laursen* teaches PL/SQL statements (detx 59)." (*Office Action*, page 13). Applicants are not sure what "detx" refers to and have determined that there is no reference in *Laursen* to the reference numeral 59. It does not appear that 59 refers to a page number, a line number, or a paragraph number. In short, the Examiner's rejection of the claim language leaves Applicants' guessing as to the basis for the Examiner's rejection and provides the Applicants with little guidance in how to respond.

Furthermore, *Laursen* does not disclose, teach, or suggest that "the specifications of object-oriented code objects comprise PL/SQL specifications for a collection of stored functions and procedures identified as a single entity." In fact, the only reference to PL/SQL in *Laursen* states that "remote procedure calls (RPC) to access services and data through the media server 100 is preferred over the use of a traditional Structured Query Language (SQL) for data access." (*Laursen*, Column 7, lines 45-49). According to *Laursen*, "[t]his is because it reduces the number of round-trip messages and provides easy to use interfaces to application services." (*Laursen*, Column 7, lines 49-51). Thus, *Laursen* actually teaches away from the use of SQL statements. Certainly there is no disclosure in *Laursen* that "the specifications of object-oriented code objects comprise PL/SQL specifications for a collection of stored functions and procedures identified as a single entity," as recited in Claim 70.

For at least these reasons, the rejections of Claim 70 should be withdrawn.

V. **Claim 73**

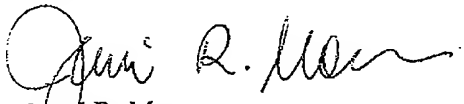
The rejection of dependent Claim 73 is improper at least because the proposed *Caron-Colby-Shan-Laursen* combination does not disclose, teach, or suggest that "displaying a dependency graph to a user, the dependency graph generated based at least in part on the dependency information tracking array." The *Office Action* acknowledges that *Caron*, *Colby*, and *Shan* do not disclose the recited claim elements and instead relies on *Laursen* for disclosure of the recited claim elements. However, *Laursen* does not at all relate to dependency graphs or to dependency information tracking arrays. *Laursen* merely discloses that "the topology of the network is described as a directed graph whose vertices are nodes (either intermediate nodes or endpoints) and whose edges are data links between nodes." (*Laursen*, Column 12, lines 40-46). Thus, while *Laursen* discusses a "graph," such disclosure is limited to a depiction of a topology of a network of servers and clients. There is no indication in *Laursen* that the directed graph is analogous to a dependency graph or a dependency information tracking array. Certainly, there is no disclosure in *Laursen* of "displaying a dependency graph to a user, **the dependency graph generated based at least in part on the dependency information tracking array**," as recited in Claim 73. Accordingly, Claim 73 is allowable over the proposed *Caron-Colby-Shan-Laursen* combination.

For at least these reasons, Applicants respectfully request reconsideration and allowance of Claim 73.

CONCLUSION

For the reasons discussed above, Applicants respectfully contend that the final *Office Action* is deficient with respect to at least independent Claims 40, 48, 65, and 72 and dependent Claims 50, 71, 73, and 74. To the extent necessary, the Commissioner is hereby authorized to charge any fees or credit any overpayments to Deposit Account No. 02-0384 of Baker Botts LLP.

Respectfully submitted,
BAKER BOTTS L.L.P.



Jenni R. Moen
Reg. No. 52,038
(214) 953-6809

Date: December 23, 2008

Correspondence Address:
at Customer No. **05073**